**Inheritance**

*Inheritance* in Java expresses an "is-a" relationship, in contrast to a "has-a" relationship, which is expressed with *composition*.

|  |  |
| --- | --- |
| **Composition**:  Use this when you would say "An object of class A ***has an*** object of class B."   * A dog has an owner. * A car has an engine. * A student has an advisor. * A line segment has a starting point and an ending point. * A ComboPolygon has an array of Polygons.   Composition expresses that one class is a *component* (a piece) of another class. | **Inheritance**:  Use this relationship to express when ***a class is a specific kind of another class.***   * A poodle is a specific kind of dog. * A racecar is a specific kind of car. * A textbook is a specific kind of book. * Inheritance expresses that one class can do everything another class can do, plus more:   + A racecar is a car that can also drive extra fast around a race track.   + A textbook is a book that is written in a specific style (and probably costs more.) |

Syntax for inheritance:

public class BaseClass {

// Whatever instance variables & methods you want

}

public class DerivedClass **extends BaseClass** {

// Whatever instance variables & methods you want.

// All variables & methods from the base class are *inherited*.

}

* The derived class inherits all the variables and methods from the base class, *just as if they had been re-declared (i.e., copy-and-pasted) in the derived class*. So objects of the derived class act just like objects of the derived class, except they might have extra abilities that are defined in the derived class.
* Variables and methods in classes may be declared *public*, *private*, or *protected*. Protected only comes into play when inheritance is involved.
* The two classes involved in this relationship are also known as the **parent class** and the **child class**.
* When a derived class inherits from a base class:
  + **Inside the derived class**, the derived class has access to all the public and protected members of the base class.
  + **Inside the derived class**, the derived class cannot access private members of the base class.
  + **Outside the derived class**, the derived class has all the same public members as the base class has, plus anything public declared in the derived class.
    - (except constructors)

**Exercise**:

* In the Parrot class, add a method for the parrot to sleep. This method should increase the parrot's energy by 5.
* Create a PetParrot class that inherits from the Parrot class. A PetParrot should be able to do everything a Parrot can, plus:
  + It should have a name that the user should be able to set.
  + It should be able to talk, which decreases its energy by 1. How will you decrease the energy?